Comparing Basic Generative and Discriminative Approaches for Named Entity Recognition

Harsh Bandhey
MS, Electrical and Computer Engineering
Duke University
harsh.bandhey@duke.edu

Maneerat Gongsiang
MS, Economics and Computation
Duke University
maneerat.gongsiang@duke.edu

ABSTRACT

In this project, we do a comparative analysis of the performance of very basic Generative Models namely Hidden Markov Models (HMMs), and Discriminative Models namely Bidirectional Long Short-Term Memory (BiLSTM) Neural Networks in a Named Entity Recognition (NER) problem statement. With the CoNLL dataset, the With Hidden Markov Model and BiLSTM Model perform 0.83 and 0.67 macro average F1-Score. We also generate synthetic data using the trained HMM generative model. For synthetic data Hidden Markov Model and BiLSTM Model perform 0.64 and 0.26 macro average F1-Score. Both approaches perform worse on synthetic data vs real data when trained on the same set, but that is most likely due to inaccurate data generation by the HMM Model.

In general, though the BiLSTM is more complex than the HMMs, even considering the synthetic data analysis, but doesn't necessarily produce higher prediction accuracy than HMM. While both approaches have their own advantages and disadvantages. The discriminative approach seems to be a better approach for a balanced dataset while the generative approach seems better for an unbalanced one for the named entity recognition problem.

Keywords

Named-entity recognition (NER), Generative Models, Discriminative Models, Hidden Markov Models, Long Short-Term Memory, and Neural Networks.

1. Introduction

Named-entity recognition (NER)

Named-entity recognition also known as (named) entity identification, entity chunking, and entity extraction is a subtask of information extraction that aims to find and classify named entities mentioned in unstructured text into pre-defined categories, such as names of people, companies, locations, and places, as well as things like amounts, percentages, time expressions, and medical codes. [1]

Generative Models

A type of statistical model known as the generative model is thought to be capable of producing new data instances. These models, which may or may not be supervised, can be applied to a variety of machine-learning tasks.[2]

The learning algorithms typically model the underlying patterns or distribution of the data points, while generative models concentrate on the distribution of specific classes in a dataset. These models make use of the idea of joint probability to produce situations in which a particular feature (x) or input and the desired outcome (y) or label (y) coexist.[2]

These models subsequently model data points and distinguish between various class labels provided in a dataset using probability estimates and likelihood.[2]

Discriminative Models

The term "discriminative model" describes a group of statistical classification models that are mostly employed for supervised machine learning. As a result of their ability to identify the borders between classes or labels in a dataset, these models are often referred to as conditional models.[2]

Discriminative models don't make any assumptions about the data points and instead distinguish classes (exactly as in the literal sense). They don't model conditional probability. However, these models are unable to provide new data points. As a result, the main goal of discriminative models is to distinguish one class from another.[2]

2. Methods

2.1 Data

2.1.1 CoNLL Dataset

The CoNLL-2003 is a named entity recognition dataset released as a part of the CoNLL-2003 shared task. The data consists of eight sets of files covering two languages: English and German. For each of the languages, there is a training set, a development set, a test set, and a large unlabeled set. For our problem, we only consider the English language dataset.[3]

The Reuters Corpus was used to obtain the English data. Reuters news articles from the months of August 1996 and August 1997 make up this corpus. Ten days' worth of data from the files indicating the end of August 1996 was taken for the training and development set. The texts for the test set were dated December 1996. September 1996 is covered by the preprocessed raw data. [3]

We process the files to create a train and test datasets where there is a text file with one word per line (a word and its label), and an empathy line separates each sentence. The original label is the combination of B/I/O and PER (person), ORG (organizations), LOC (location), and MISC (Miscellaneous), but we focus only on the entities by treating three different types of each entity as the same. In other words, I-ORG, O-ORG, and B-ORG are defined as ORG. The training set contains 204567 entity words, while the test set contains 98316 entity words.

2.1.2 Synthetic Dataset

We also generate a synthetic dataset using the generative model trained on the CoNLL Dataset. To generate the said dataset we generate the same number of sentences as the real set but between a random length between 10 to 25 using the trained HMM Model. This approach is likely to affect synthetic data performance as this arbitrarily reduces the number of token words in the dataset. We do this process by using the random_sample function from the NLTK package, which randomly chooses the starting state and probability and then samples the state transition accordingly. This creates word sequences with their entity according to the HMM's probability distribution. [5]

2.2 Models

2.2.1 Hidden Markov Model

We adopted the Hidden Markov Model (HMM) to identify the entities based on a known sequence of words. The model assumes the probability of transition from one state to another only depends on the current state (Markov chain's assumption). Specifically, the HMM defines:

- The set of states: the entities are PER, ORG, LOC, MISC, and O
- The transition probabilities: the probability of transition from a given entity to each entity
- The output observation alphabet: the set of words that is a probabilistic function of the state or entities in this case.
- The output probability matrix: the probability of observing each word in a given entity
- The initial state distribution: the probability of starting in each entity

To train the model, we encode the labeled training dataset using one-hot encoding and calculate the joint probability of each word. Then, the highest probability state sequence is tagged to the sequence and returns the optimal state sequence through the HMM. The model optimizes the individual entity by considering the combination of entities for a larger group of words using the Viterbi algorithm with dynamic programming. As a result, the Viterbi algorithm gives the highest probability of an entity sequence that maps to a word sequence. To learn HMM parameters from the training data, supervised learning (MLE) is adopted to maximize the joint probability of the word and entity sequences. [5]

2.2.2 BiLSTM Neural Network

To train a neural network, we use the following methodologies below.

Many to Many Architecture: Neural network models work with graphical structure. Therefore we will first need to design the architecture and set input and out dimensions for every layer. RNNs are capable of handling different input and output combinations. We will use "many to many" architectures for this task. Our task is to output a tag for a token ingested at each time step.

Glove Embeddings: Glove embeddings will give embeddings for the current token. GloVe embeddings

are vector representations for words. These representations showcase interesting linear substructures in the word vector space. We use the pre-trained version of this where Training is performed on aggregated global word-word co-occurrence statistics from a corpus. Specifically, we use embeddings from the Twitter 2B tweets corpus with 27B tokens with an embedding dimension of 100. [6]

Padding Methodology: The LSTM layers accept sequences of the same length only. Therefore, every sentence represented as integers must be padded to have the same length. We will work with the max length of the longest sequence and pad the shorter sequences to achieve this.

The Model

For our architecture, as shown in Figure 1, we are primarily working with 4 layers (input, embedding, BiLSTM, LSTM layers) and the 5th layer, which is the TimeDistributed Dense layer, to output the result.

Layer 1 - Input Layer: Takes as input token a list of token indexes with the specified maximum length of the padded sequences.

Layer 2 - Embedding layer: the embedding layer will transform each token into a vector of 100 dimensions glove Embedding representation.

Layer 3 - Bidirectional LSTM: Bidirectional LSTM takes a recurrent layer as an argument. Since this is a bidirectional LSTM, we will have forward and backward outputs. We combine these outputs using concatenation.

Layer 4 - LSTM Layer: An LSTM network is a recurrent neural network that has LSTM cell blocks in place of our standard neural network layers. These cells have various components called the input gate, forget gate, and output gate.

Layer 5 - TimeDistributed Layer: We are dealing with Many to Many RNN Architecture, where we expect output from every input sequence. The TimeDistributeDense layers allow Dense (fully connected with softmax activation) operation across every output over every time step. Not using this layer will result in one final output.

Training and Output

We convert the target variable as a one-hot encoded vector and use categorical cross entropy as a loss. Thus we get a logit probability of the tags as an output which we argmax to get the result/prediction. Training is done with a Gradient Descent with Adam Optimizer for 30 epochs with a batch size of 256. Figures 1 and 2 show descending loss plots for the models accordingly.

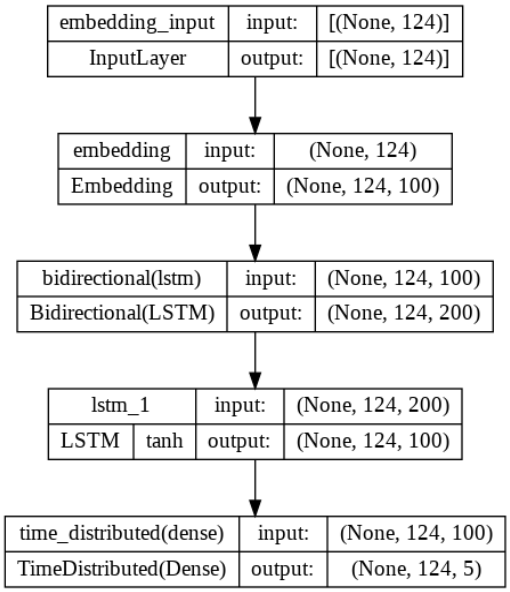


Figure 1. BiLSTM Neural Network Architecture

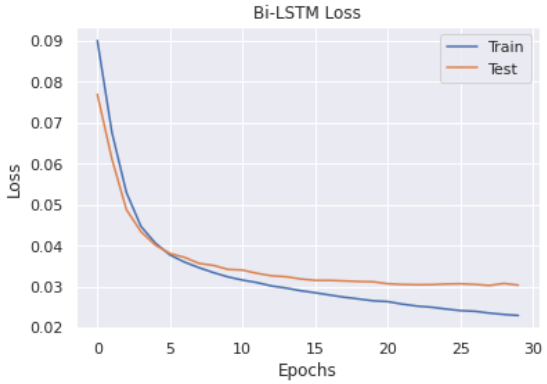


Figure 2. BiLSTM Model Loss Plots (Real Data)

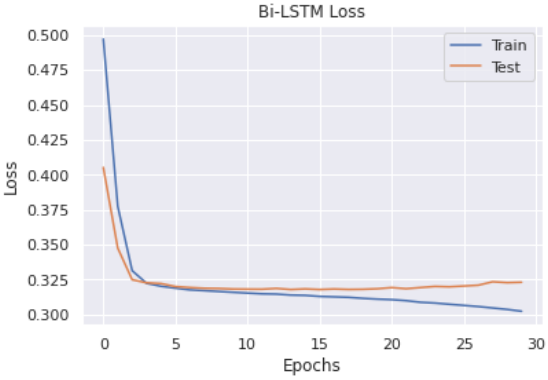


Figure 3. BiLSTM Model Loss Plots (Synthetic Data)

3. Results

3.1 Quantitative Assessment

Hidden Markov Model: The overall quality of prediction for real-world data and synthetic data is not significantly different. That is, the weighted average F1-score is 0.95 and 0.90 for real-world and synthetic data, respectively. The F1-score for both data types is relatively high, meaning the overall prediction quality is good. In both cases, the overall good prediction comes from a good precision and recall score of roughly 0.90 (Tables 1 and 3).

However, the quality of predictions is not always high for some entities, reflected by the f1-score for ORG and PER from real-world data models and most entities from synthetic data that are pretty low. For real-world data, the f1-score for ORG is only 0.74, relatively low compared to LOC and ORG. The poor performances of ORG and PER prediction are shown in the recall score, where the value is below 0.70. That is, all words that are actually ORG and PER are correctly predicted less than 70% of the time. This is reasonable as it is challenging to predict the entity of words that are quite unique such as the person's name and organization (Table 1). This is also true for the case of synthetic data.

Moreover, the overall prediction quality with synthetic data is dominated by the performance of words that are not nouns (O). Meanwhile, the prediction quality for LOC, MISC, ORG, and PER is quite low, indicated by the relatively low f1-score. The poor performance for most entities is shown in the precision score below 0.65. That is, out of all words that are predicted to be LOC, MISC, ORG, and PER are correct 57%, 46%, 59%, and 64% of the time, respectively. The small number of observations likely accounts for this poor performance in the synthetic data case (Table 3).

BiLSTM Neural Network: The overall quality of prediction for real-world data is much better than synthetic data. The weighted average F1-score is 0.91 and 0.79 for real-world and synthetic data, respectively. The better performance of real-world data is due to both precision and recall scores, while the poor performance of synthetic data is primarily due to the low precision score. (Tables 2 and 4)

Similar to HMM, the quality of predictions is not always high for some entities, especially for synthetic data. The FC-score for LOC and MISC is very close to zero, which means the models produce a poor prediction when the words are LOC or MISC. (Table 4). One observation here is that the BiLSTM requires many data inputs to get an overall prediction as good as HMM and performs especially poorly on classes with low no of samples.

	precision	recall	f1-score	support
LOC	0.80	0.88	0.84	3665
MISC	0.75	0.86	0.80	1903
O	0.99	0.95	0.97	84102
ORG	0.67	0.85	0.74	3611
PER	0.68	0.89	0.77	4516
accuracy			0.94	97797
macro avg	0.78	0.89	0.83	97797
weighted avg	0.95	0.94	0.95	97797

Table 1. Classification Report, HMM Model with Real Data

	precision	recall	f1-score	support
LOC	0.67	0.46	0.55	4019
MISC	0.51	0.43	0.47	2186
O	0.95	0.97	0.96	81082
ORG	0.61	0.54	0.58	4588
PER	0.76	0.82	0.79	5922
accuracy			0.91	97797
macro avg	0.70	0.65	0.67	97797
weighted avg	0.90	0.91	0.91	97797

Table 2. Classification Report, LSTM Model with Real Data

	precision	recall	f1-score	support
LOC	0.46	0.75	0.57	265
MISC	0.39	0.57	0.46	197
O	0.97	0.93	0.95	9866
ORG	0.55	0.64	0.59	466
PER	0.58	0.72	0.64	497
accuracy			0.90	11291
macro avg	0.59	0.72	0.64	11291
weighted avg	0.92	0.90	0.90	11291

Table 3. Classification Report, HMM Model with Synthetic Data

	precision	recall	f1-score	support
LOC	0.25	0.01	0.01	4516
MISC	0.00	0.00	0.00	2549
O	0.86	0.99	0.92	95958
ORG	0.31	0.20	0.24	5287
PER	0.31	0.07	0.12	6004
accuracy			0.84	114314
macro avg	0.35	0.25	0.26	114314
weighted avg	0.77	0.84	0.79	114314

Table 4. Classification Report, LSTM Model with Synthetic Data

3.2 Qualitative Assessment

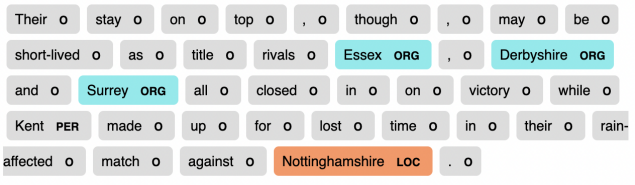
3.2.1 Real Dataset

Figures 4 to 6 illustrate examples of predictions from HMM and BiLSTM with real-world data. The original NER is “Their stay on top, though, may be short-lived as title rivals Essex, Derbyshire, and Surrey all closed in on victory while Kent made up for the lost time in their rain-affected match against Nottinghamshire.” There are five nouns which are all categorized in the ORG group. While the HMM perfectly predicts the entities of all nouns, the LSTM produces two wrong results: Kent as a PER instead of ORG and Nottinghamshire as a LOC instead of ORG. The poor performance of LSTM is reasonable as the words ‘Kent’ can be a person and ‘Nottinghamshire’ is very likely to be the location. However, since the HMM considers the probability of a word based on the dataset, its performance can be more accurate than LSTM.

Figure 4: Original NER

Figure 5: HMM Prediction

Figure 6: LSTM Prediction



3.2.2 Synthetic Dataset

Figures 7 to 9 illustrate examples of predictions from HMM and BiLSTM with synthetic data. The original NER is “Finally, the Two of, (Friday AND is who the Jerry Bevan David 0.38 Hungary of the surprise” There are four nouns: ' Jerry’, ‘Bevan’, ‘David’, are categorized as PER, and ‘Hungary’ is LOC. As expected, the HMM correctly predicts the entities of all nouns, while the LSTM predicts every word as ORG. This result should account for the fact that the synthetic data is generated based on the probability from the HMM model.

Figure 7: Generated NER

Figure 8: HMM Prediction

Figure 9: LSTM Prediction

3.3 Analysing Results

The very poor prediction performance of LSTM with synthetic data is surprising. That is, for most of the entities: LOC, MISC, and PER, their prediction performance is very close to zero. As we observe from the example in figure 9, the sentence generated from the HMM is relatively unstructured and unreasonable. This might be hard for LSTM to learn from the dataset, leading to low precision and recall scores. This is especially true while using Glove embeddings trained from a general language data corpus.

3.4 Comparison

3.4.1 Quality and correctness

Overall, the HMM and BiLSTM work best with real-world data, reflected by the high weighted average prediction scores: 0.95 and 0.91, respectively. However, when considering synthetic data generated from HMM, the prediction score from BiLSTM is very low: 0.90 and 0.79, respectively.

With real-world data, HMM predicts ORG and PER relatively unsatisfactorily. This might be due to the overlapping of words across entities. For example, the word 'Kent' could be a person's name (PER) and location (LOC) with similar probability. However, BiLSTM produces poor predictions, particularly for MISC. This might be because the MISC, referred to as a Miscellaneous entity: events,

nationalities, products, or works of art, is too varied, making it hard to learn.

With synthetic data, HMM correctly predicts entities compared to BiLSTM, as expected. It is reasonable that BiLSTM provides very poor prediction as with unstructured sentences (generated from HMM), it is hard to learn an entity for a specific word from the context. As a result, the BiLSTM rarely predicts the correct entity, especially for MISC, which is supposed to rely heavily on the context.

3.4.2 Data and computational requirements

The input data for the HMM model is pairs of a word and entity while in the LSTM model, a list of sequences is transformed into a 2D NumPy array, a numerical representation of a word. In addition, the entities in LSTM are represented using one-hot encoding used to represent a word. Moreover, LSTM being a deep learning algorithm learns the latent representations of the word vector space and thus needs much more data to perform in a meaningful way.

In terms of execution time, the LSTM takes a longer time than the HMM model. The HMM model could take up to 2 minutes for prediction while the LSTM model could take up to less than a minute. Even though the execution time for LSTM is shorter, it is due to the difference in inference and training. In general, the HMMs are much simpler than the LSTM model. The LSTM requires more sophisticated models and takes longer to converge. In turn, it needs GPU for training and further GPU costs with subsequent inference runs. Additionally, the HMMs rely on strong assumptions where the State transitions only depend on the current state, which reduces a lot of computation tasks.

3.4.3 Interpretability

As mentioned earlier, the HMMs are simpler and more transparent models, therefore the HMMs are more understandable and interpretable. That is, the HMMs are characterized by five main elements: the number of hidden states, transition probability, a sequence of observations, the observation likelihood matrix, and initial state probability. The outcomes, and sequence of entities, are generated by maximizing the joint probability of the terms and labels. These probabilities are estimated by counting the transitions between label entities according to observations in the training data. Then we compute the posterior probability of being in an entity given a word from forward and backward probabilities. In contrast to the HMMs, the LSTM Model contains many layers and the internal structure of the LSTM-based model can be user-defined and quite complicated. Moreover, LSTMs are not interpretable, it is a "deep learning" based and learns features that are not clearly understandable to human intuition. We cannot know what it sees to say why it chose that label.

4. Discussion

While Generative Models can also produce new data points, unlike discriminative models. However, they also have a critical flaw: These models are significantly impacted by the presence of outliers in the dataset.

Discriminative models perform better when there are outliers in the dataset than generative models because they are more resistant to outliers. These models do have one significant flaw, though, and that is the misclassification problem, which refers to classifying a data piece incorrectly.

5. Conclusion

In conclusion we've trained both simple generative and discriminative models, HMMs and BiLSTMs respectively. The BiLSTM though more complex than the HMMs produces similar or less prediction accuracy than HMM, especially considering the synthetic data analysis. While both approaches have their own advantages and disadvantages. We now know the advantages and disadvantages of both models and how the generative approach seems to be a better approach for the named entity recognition problem when a problem of class imbalance seems apparent.

6. Acknowledgments

We would like to express our gratitude to Dr. Patrik Wang to give us the opportunity to explore this project, for his help throughout the course, and for teaching us new methodologies in Natural Language Processing.

7. Code and Data Availability

The code base for this project can be entirely found in the following Google Drive link:
https://drive.google.com/drive/folders/1UIT3uafYSS8AXX4Y-Q5YFCaV-YLcHUb5

The following Google Colab notebook also runs all the code for the project independently:
https://colab.research.google.com/drive/11eRt4IHMlJX3KfTlR2UjW0ED59AuHAMI

8. References

[1] Li, Jing, et al. "A survey on deep learning for named entity recognition." IEEE Transactions on Knowledge and Data Engineering 34.1 (2020): 50-70.

[2] Goyal, C. (2021, July 19). Deep Understanding of Discriminative and Generative Models in Machine Learning. Analytics Vidhya. https://www.analyticsvidhya.com/blog/2021/07/deep-understanding-of-discriminative-and-generative-models-in-machine-learning/

[3] Sang, Erik F., and Fien De Meulder. "Introduction to the CoNLL-2003 shared task: Language-independent named entity recognition." arXiv preprint cs/036050 (2003)

[4] Nair, S. (2022, September 21). Named-Entity Recognition using Keras Bi-LSTM | Towards Data Science. Medium. https://towardsdatascience.com/named-entity-recognition-ner-using-keras-bidirectional-lstm-28cd3f301f54

[5] Cohn, T., Blunsom, P., Tresoldi, T., Bird, S., Frazee, J., & Xu, S. (2022, March). *Source code for nltk.tag.hmm*. NLTK. Retrieved December 10, 2022, from https://www.nltk.org/_modules/nltk/tag/hmm.html

[6] Pennington, Jeffrey, Richard Socher, and Christopher D. Manning. "Glove: Global vectors for word representation." Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP). 2014.

[7] Waseem, Omer. "omerwase/Named-Entity-Recognition: NER using Hidden Markov Model, Conditional Markov Model and Bidirectional Long Short Term Memory NN." *GitHub*, April 2018, https://github.com/omerwase/Named-Entity-Recognition.